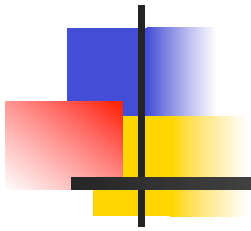
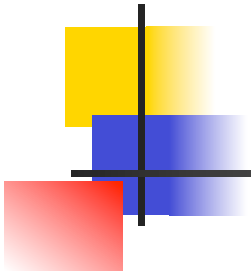


Systèmes concurrents et répartis

Algorithmie
File d'attente répartie

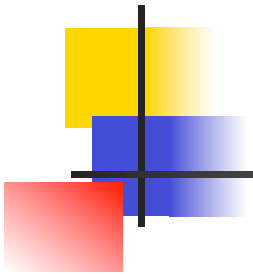
Laurent Pautet
Laurent.Pautet@enst.fr

Version 5.0



Etude de cas

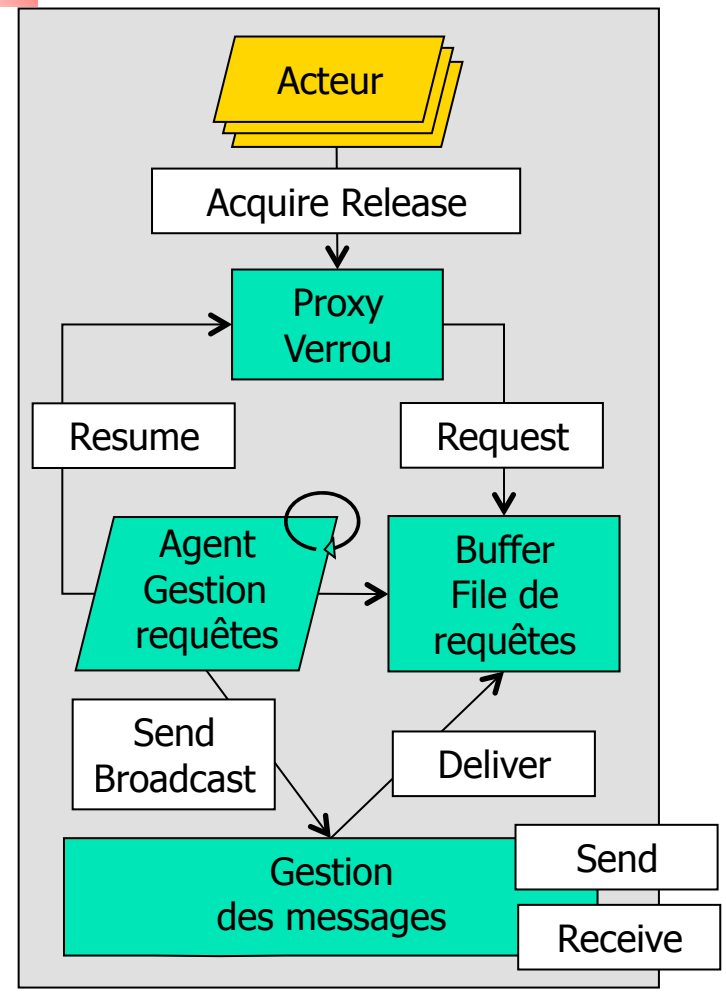
- Des étudiants s'échangent un album de photos et chacun souhaite y rajouter de nouvelles photos.
- Pour ne perdre aucune photo, les étudiants doivent modifier l'album en exclusion mutuelle
- L'accès à l'album s'obtient par un verrou réparti
- Nous étudions **l'accès** à cet album et donc la gestion de ce verrou réparti selon différentes approches
- Nous n'étudions pas les modifications sur les données de l'album

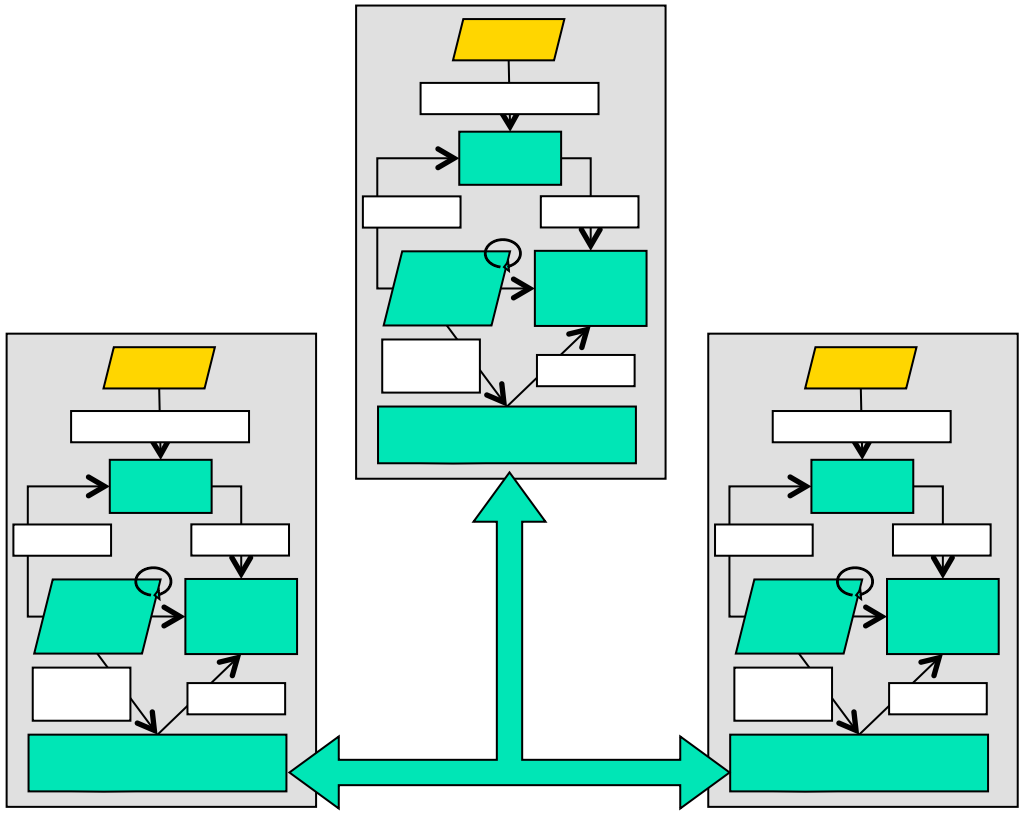


Verrou réparti

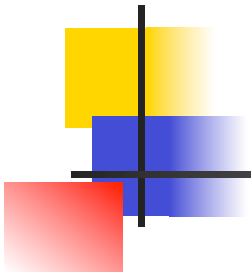
- L'accès est géré par un verrou réparti entre étudiants
- Chaque étudiant se compose de deux activités :
  - un acteur qui utilise un proxy du verrou, interface d'un verrou supposé local qui cache la complexité d'un verrou réparti
  - Un agent qui traite les demandes du proxy, garantit l'exclusion mutuelle et interagit avec les autres agents
- Le proxy offre la même interface que l'objet initial
- Le proxy transforme les fonctions d'un verrou local en des requêtes complexes auprès de l' « agent »
- L'agent applique un algorithme qui assure la cohérence du verrou et débloque le proxy et donc l'acteur

Cas d'étude
Architecture





L. Pautet

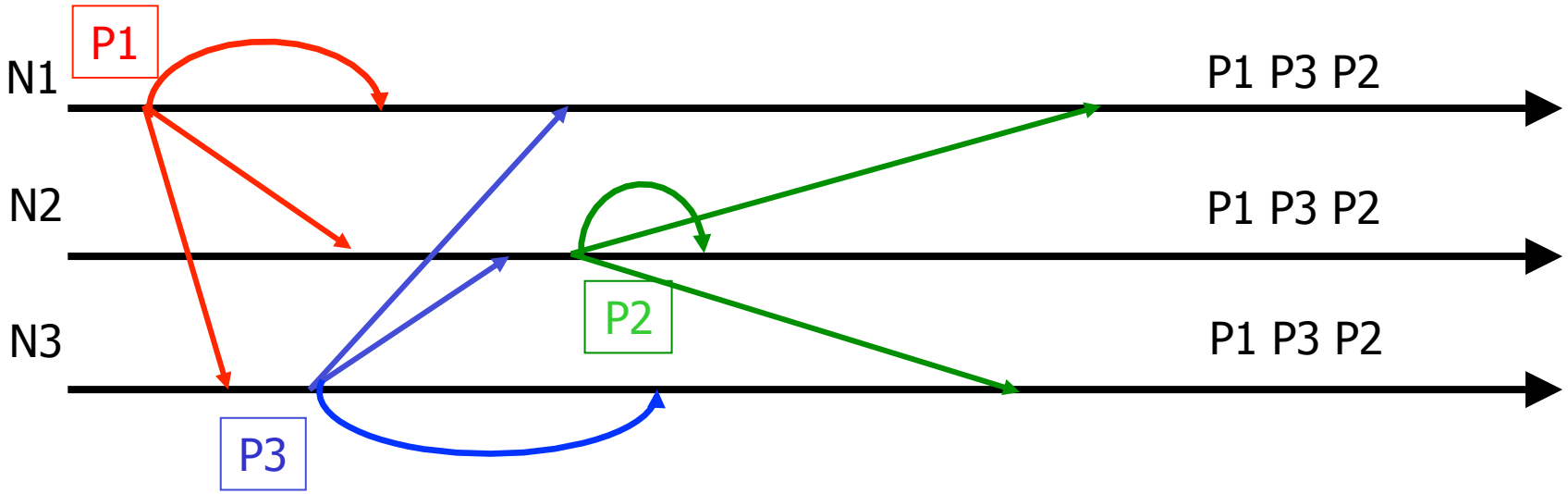


Exclusion mutuelle avec diffusion de groupe

- Une première approche pour gérer un verrou en réparti consiste à utiliser la diffusion de messages à un groupe
- Un nœud peut demander et relâcher l'accès en diffusant ses requêtes auprès du groupe
- La diffusion de groupe doit assurer certaines propriétés pour garantir la cohérence de l'exclusion mutuelle
- Une diffusion émise logiquement après une autre est reçue dans cet ordre sur tous les nœuds (ordre causal)
- Deux diffusions concurrentes arrivent dans le même ordre sur tous les nœuds (ordre total)

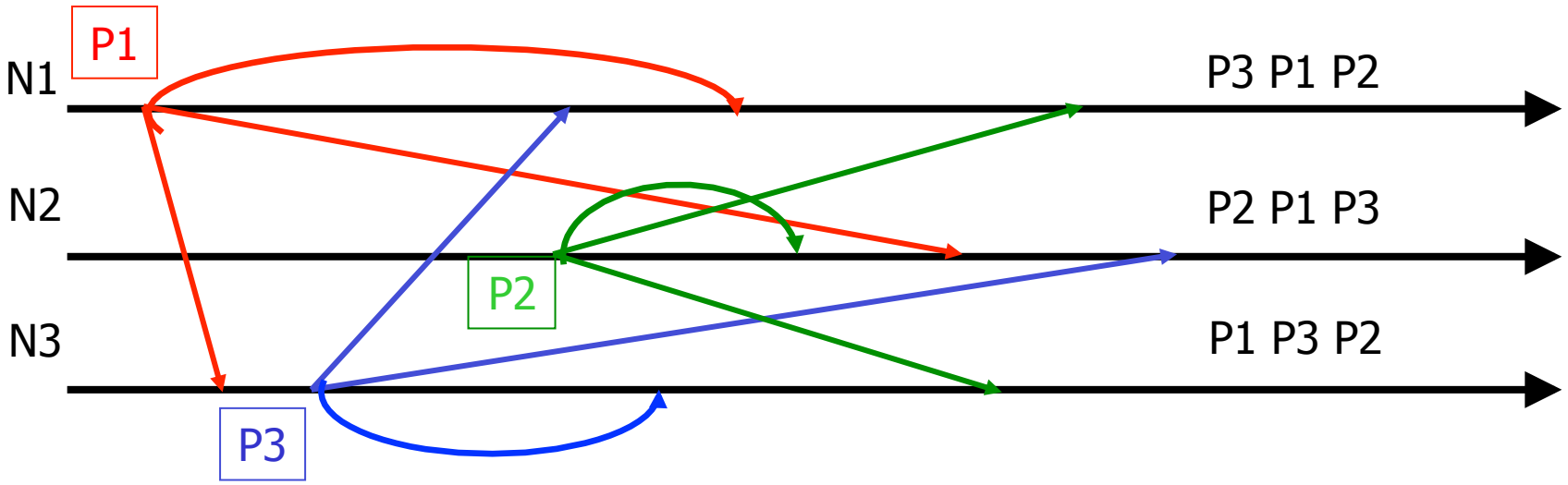
Diffusion idéale

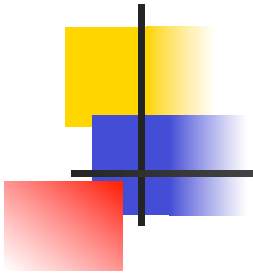
- Pour les nœuds N1, N2 et N3
- Pi signifie P(S) par le nœud Ni
- Vi signifie V(S) par le nœud Ni
- Les requêtes P arrivent dans le même ordre sur tous les nœuds



Diffusion réelle

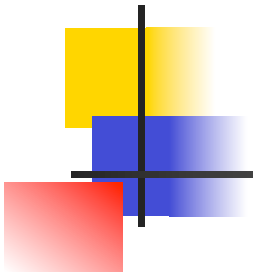
- P1 arrive en N3 avant P3, mais N1 reçoit P3 avant P1
- P1 et P2 ne sont pas reçus dans le même ordre entre N1 et N2
- P1, P2, P3 ne sont reçues dans le même ordre sur aucun nœud





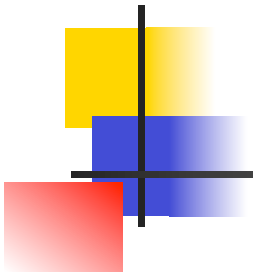
Diffusion de groupe

- Principes et Définitions
- Coordonnateur central
- Anneau virtuel avec jeton
- Horloges de Lamport et de Mattern
- Enquête généralisée
- Diffusion par horloge vectorielle



Causalité

- Causalité élémentaire
  - Pour tout M, émission (M) -> réception (M)
  - « -> » signifie « précède»
- Causalité répartie (ordre partiel)
  - A « précède causalement » B si A et B sont des événements produits dans cet ordre sur un même nœud
  - A « précède causalement » B si A est l'événement d'émission d'un message M par le nœud N1 et B l'événement de réception du message M sur N2
  - La relation de causalité répartie est la fermeture transitive
  - Si A -> B et B -> C alors A -> C



Ordre causal des messages

Pour tout M1 émis de N1 vers N3

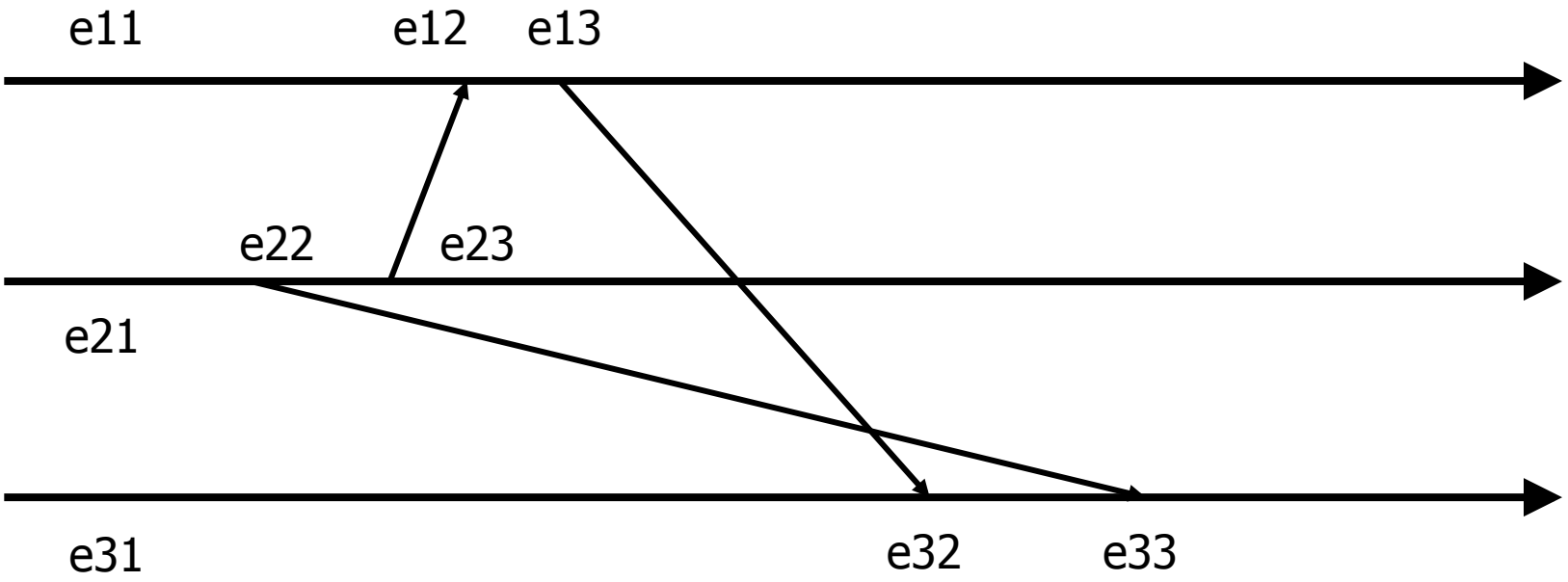
Pour tout M2 émis de N2 vers N3

émission (M1) -> émission (M2)

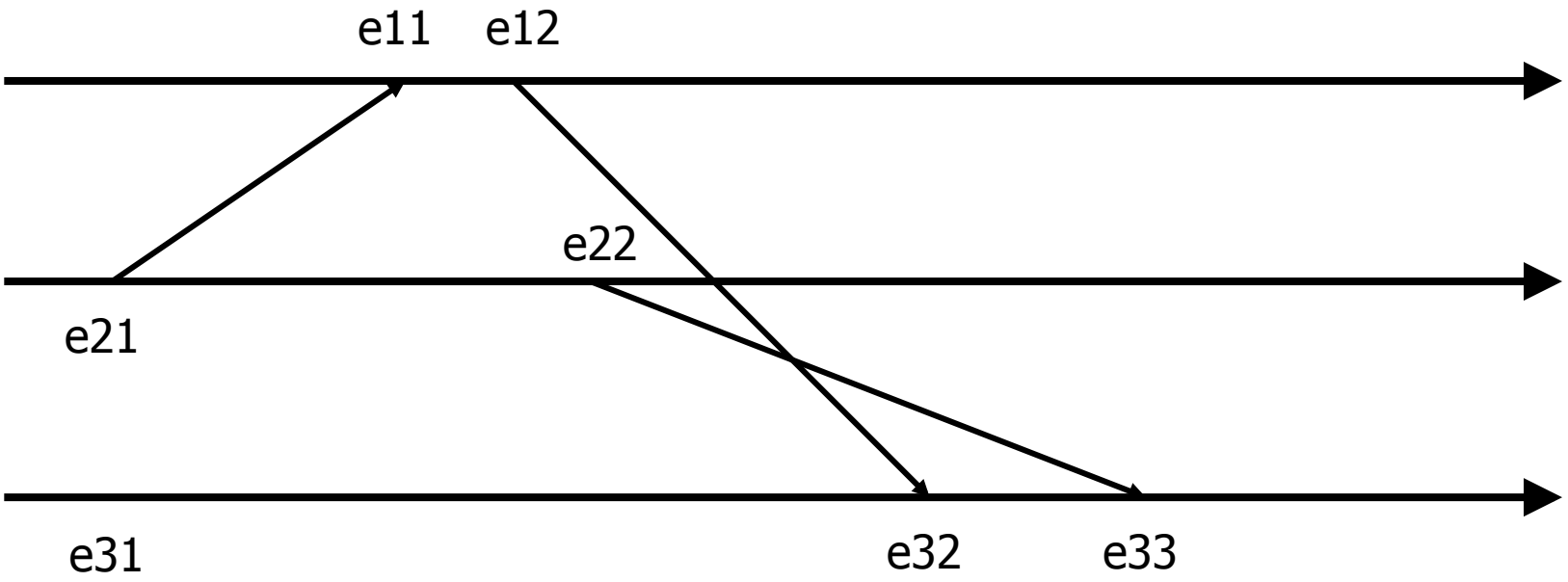
=>

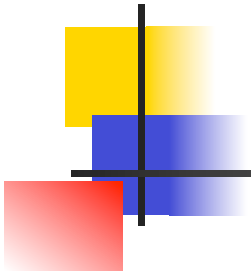
réception (M1) -> réception (M2)

Non-respect de causalité



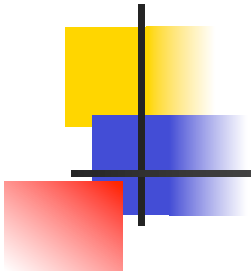
Respect de causalité





Diffusion avec ordre causal

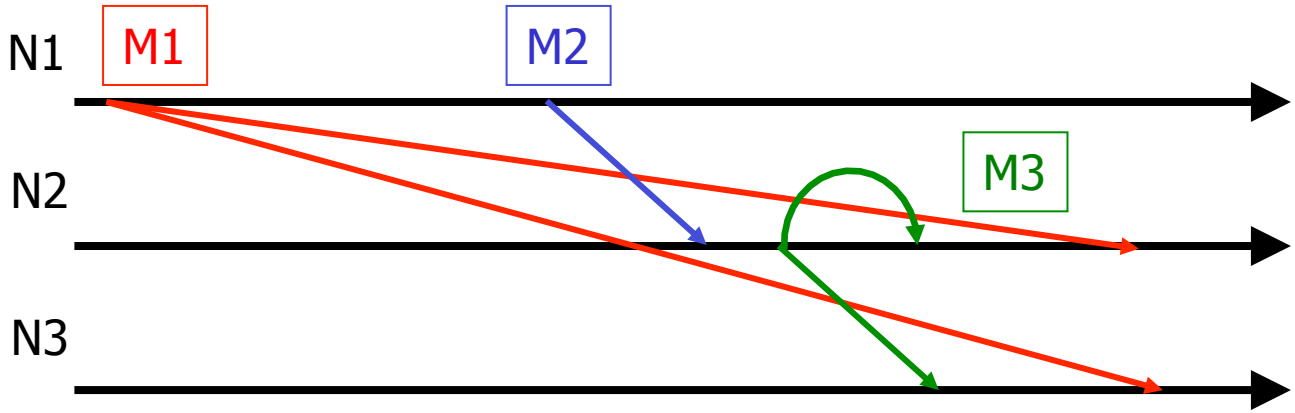
- Diffusion respectant l'ordre local
  - Pour deux diffusions successives du même processus, les messages sont délivrés dans le même ordre chez chaque destinataire
- Diffusion respectant l'ordre causal
  - Toute suite de diffusions de message en relation de causalité implique la délivrance des messages chez les destinataires dans la même relation de causalité

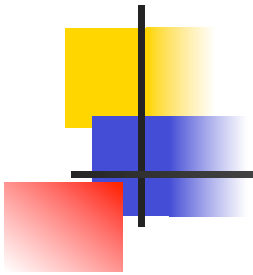


Diffusion avec ordre total

- Diffusion avec ordre total simple
  - Si deux diffusions ont lieu concurremment vers le même groupe, alors les deux messages sont délivrés aux destinataires dans le même ordre, éventuellement différent de celui d'émission.
- Diffusion avec ordre total causal
  - L'ordre total respecte aussi la relation de causalité des messages.

Ordre total ≠> Ordre causal





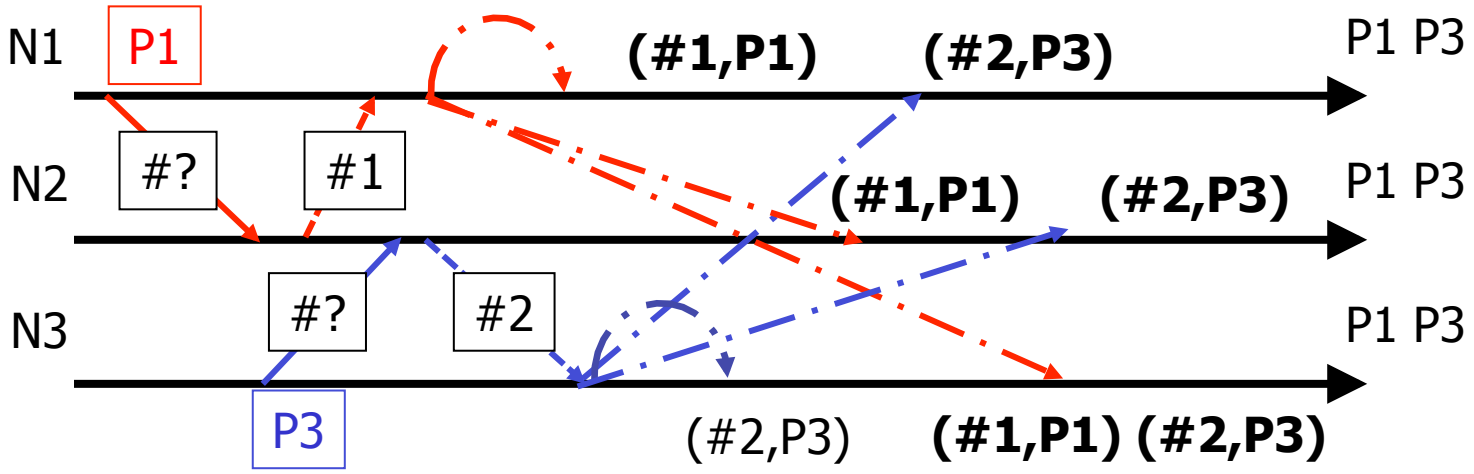
Différents séquenceurs

Ordre total ou causal ?

- Un nœud N est choisi pour jouer le rôle de séquenceur.
- Lorsqu'un nœud diffuse un message, il interagit avec le séquenceur pour obtenir le numéro de séquence suivant
- Les récepteurs trient les messages par ordre de numéro
- S'ils n'ont pas de numéro, ils sont mis en attente
- S'il manque un numéro, on attend le message manquant
- On **délivre** un message s'il a le + petit numéro attendu
- Les canaux doivent être FIFO
- En cas de panne de N, on élit un nouveau séquenceur et on reprend les émissions là où N les a laissés.

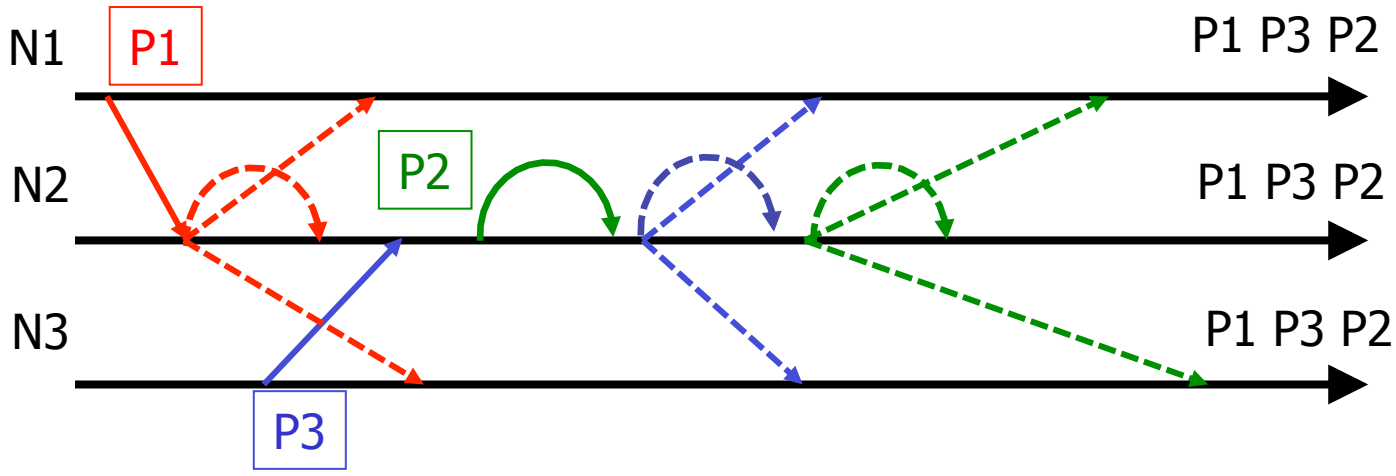
Différents séquenceurs
Unicast - Unicast

- N2 est le séquenceur (goulot d'étranglement, point de défaillance)
- N1 demande à N2 un numéro de séquence qui lui répond
- N1 diffuse le message avec son numéro de séquence
- Les nœuds trient les messages par numéro
- S'il manque un numéro, on attend qu'il arrive (#2 pour N3)
- On **délivre** à l'application le message de plus petit numéro



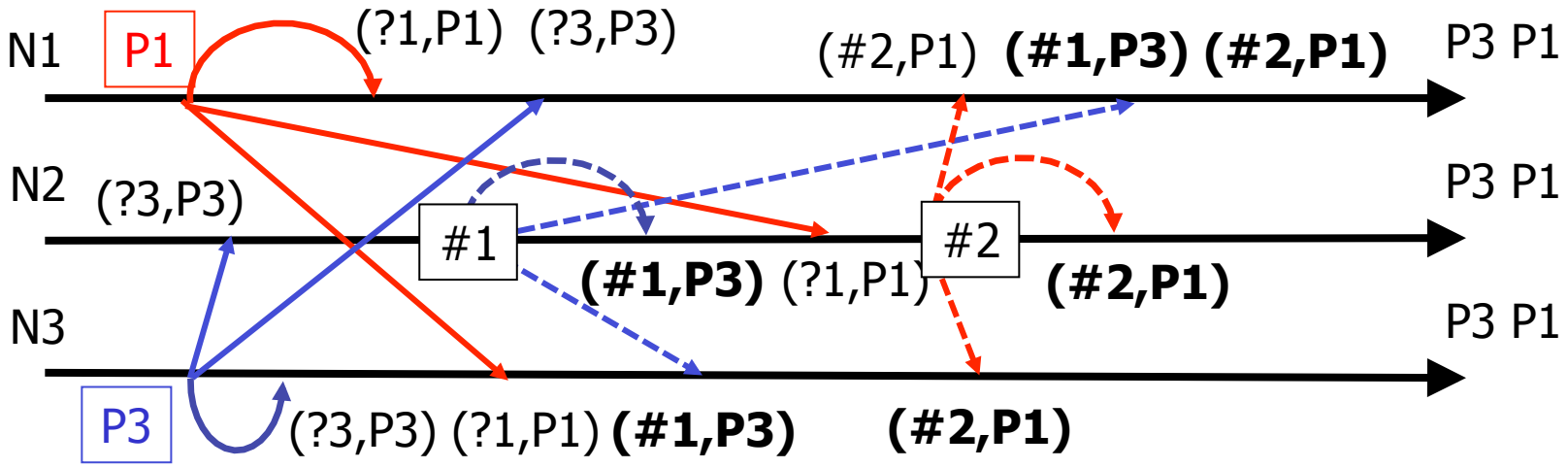
Différents séquenceurs
Unicast - Broadcast

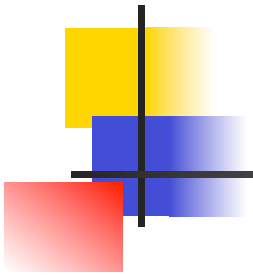
- N2 est le séquenceur (goulot d'étranglement, point de défaillance)
- N1 envoie le message P1 vers N2 qui le diffuse au groupe
- N2 le diffuse avec un numéro de séquence (mais ici FIFO)
- N2 est un fort goulot d'étranglement (il diffuse les messages !)



Séquenceur
Broadcast - Broadcast

- N2 est le séquenceur (goulot d'étranglement, point de défaillance)
- N1 diffuse son message
- N2 reçoit le message, calcule un numéro de séquence et le diffuse
- Les nœuds trient les messages par numéro
- On attend d'avoir le numéro pour le message du nœud concerné
- On **délivre** le message de plus petit numéro successif





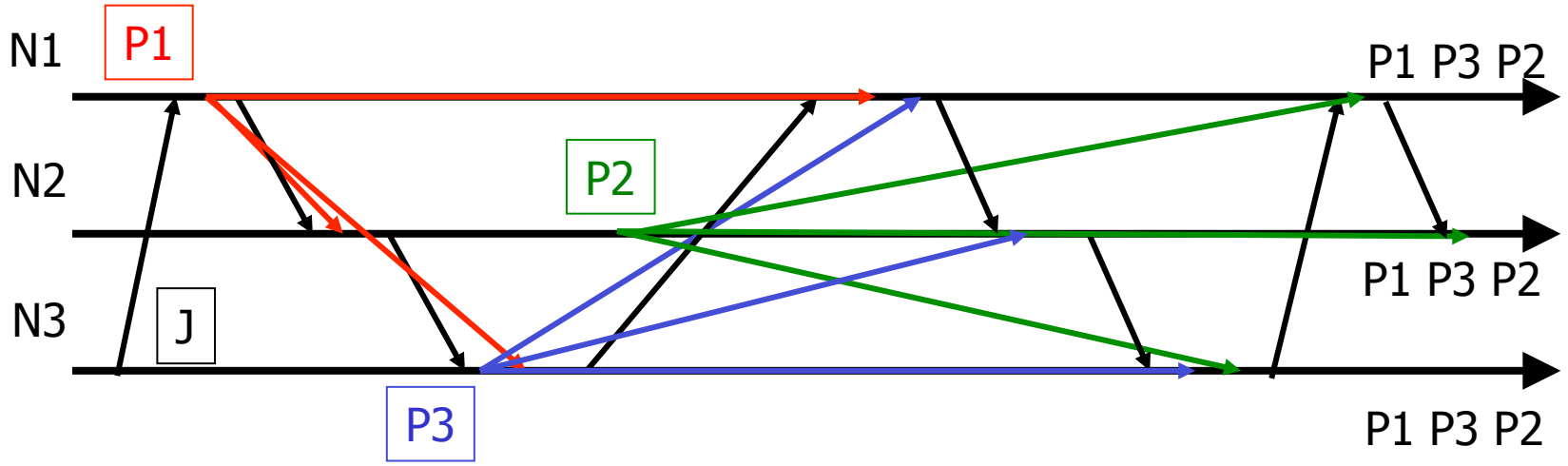
Anneau virtuel avec jeton

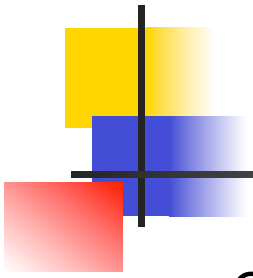
Ordre total ou causal ?

- Un jeton tourne sur un anneau virtuel et transporte les messages à diffuser.
- A la réception du jeton sur le nœud N_i
  - Délivrer en local dans l'ordre M_{N_i}, $M_{N_{i+1}}$,…, $M_{N_{i-1}}$
  - Envoyer à N_{i+1} le jeton avec $M_{N_{i+1}}$,…, $M_{N_{i-1}}$,M'_{N_i}
- Le jeton tourne de manière continue et sollicite des nœuds qui n'ont rien à émettre.

Anneau virtuel avec jeton

- N1 ajoute P1 au jeton lors de son passage
- N1 retire et délivre P1 lors du passage suivant du jeton
- La solution produit de nombreux messages (vides)
- La latence est importante si les nœuds sont nombreux





Datation
Horloges de Lamport

- Chaque nœud N_i dispose d'une horloge H_i initialisée à 0.
- Un nœud N_i incrémente H_i lors d'un événement local.
- Un nœud N_i estampille tout message émis avec H_i.
- Le récepteur N_j du message estampillé H_i recale son horloge H_j par $\max(H_i, H_j) + 1$

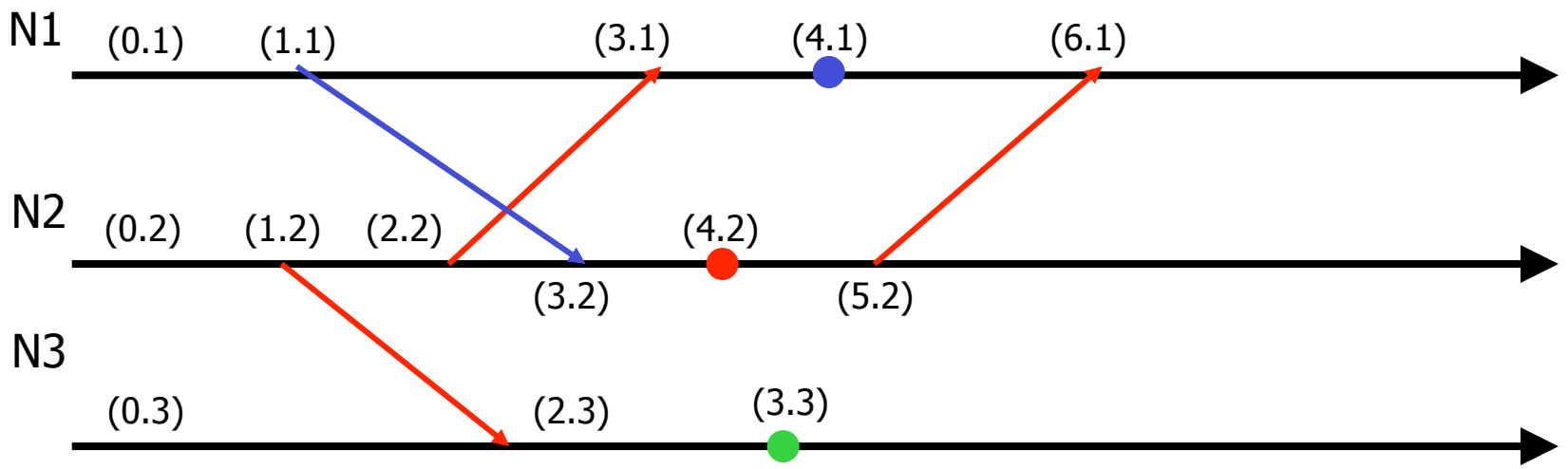
$$(a_i \Rightarrow b_j) \Leftrightarrow (H(a) < H(b)) \Leftrightarrow (H_i < H_j) \text{ ou } (H_i = H_j \text{ et } i < j)$$

- L'ordre partiel introduit préserve la dépendance causale.
- L'ordre total s'obtient en incluant le numéro du nœud (Hi, i).
- L'ordre obtenu n'établit pas une relation de causalité

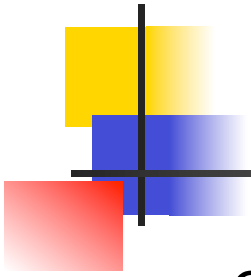
$$(a \rightarrow b)_{Lamport} \Rightarrow (H(a) < H(b))_{Causal}$$

$$(H(a) < H(b)) \Rightarrow \neg\, (b \rightarrow a)$$

Horloges de Lamport



Datation
Horloge de Mattern



- Chaque nœud N_i définit une horloge $V_i(1..N)$ initialisée à 0.
- Un nœud N_i incrémente $V_i(i)$ pour tout événement.
- Un nœud N_i estampille tout message M émis avec V_i.
- Le récepteur N_j de M estampillé Vm recale son horloge V_j . $\forall\ k \in 1..N$, $V_j(k) = \max(Vm(k),V_j(k))$

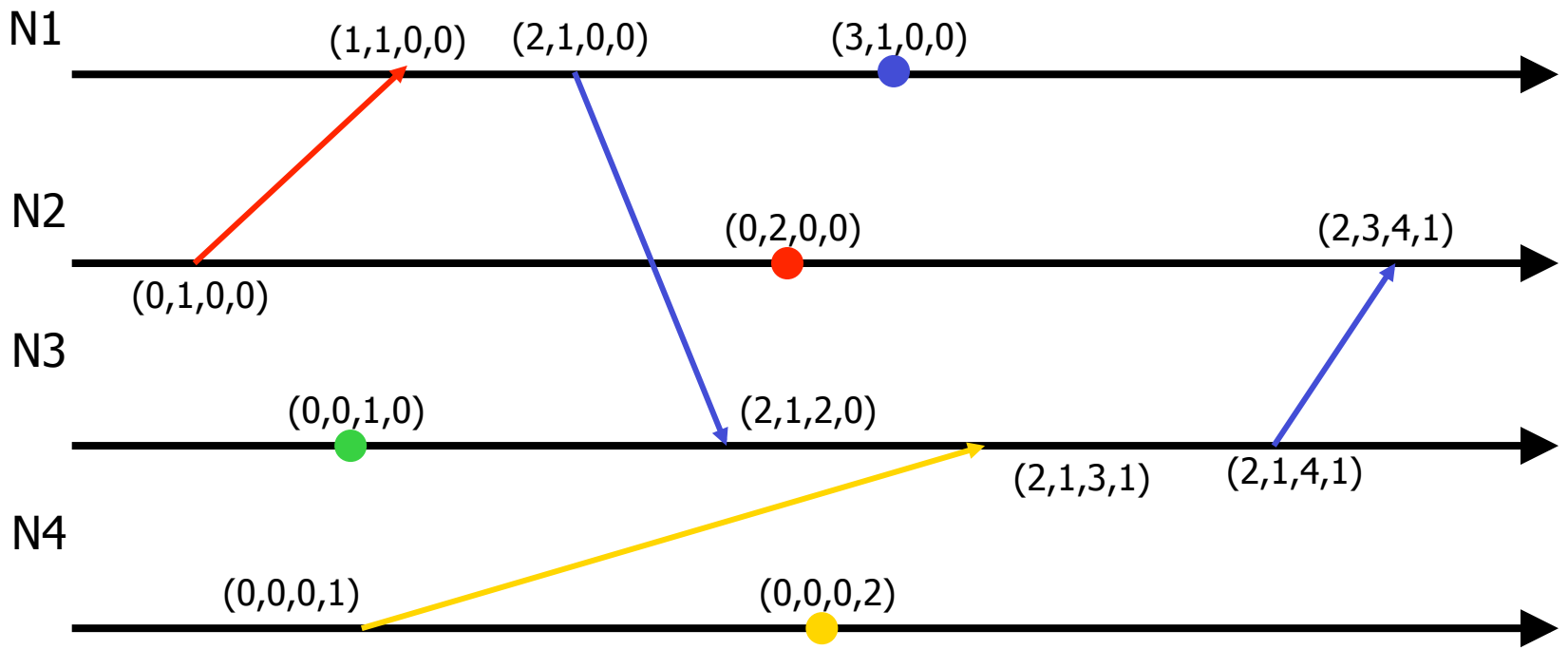
$$(A \leq B) \Leftrightarrow (\forall\ k \in 1..N,\ A(k) \leq B(k))$$

$$(A < B) \Leftrightarrow (A \leq B \text{ et } A \neq B)$$

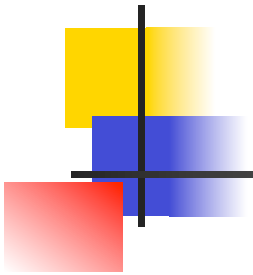
- Cet ordre partiel représente exactement la dépendance causale.
- A(i) constitue le nombre d'événements qui précèdent A et qui ont eu lieu sur le nœud N_i .
- $\sum$ A(k) constitue le nombre d'événements qui précèdent A.

Horloges de Mattern

ou horloges vectorielles



Laurent Pautet



Enquête généralisée
Ricart - Agrawal

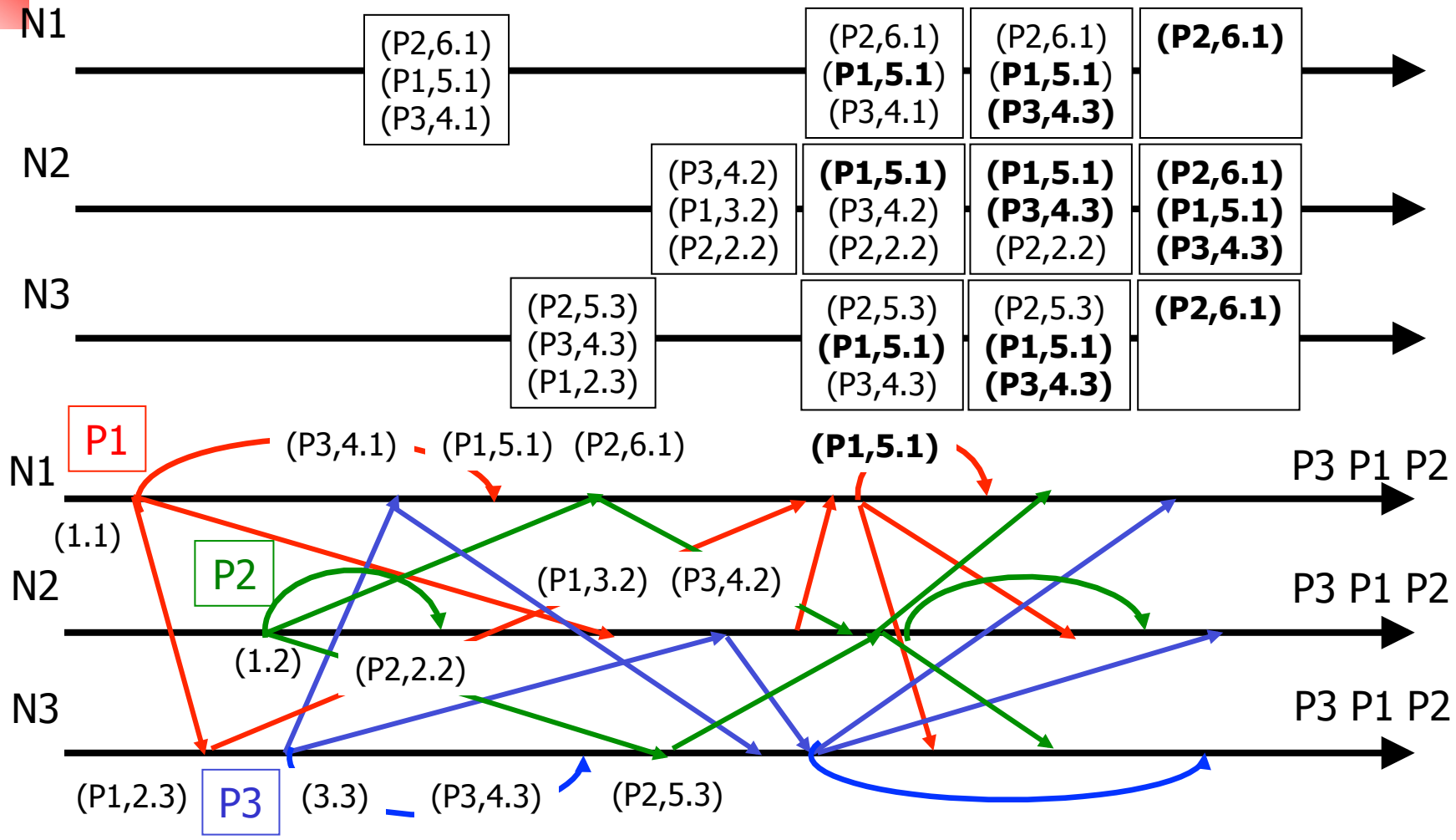
- Tout nœud N maintient une liste de messages reçus ainsi qu'une estampille de Lamport S_N.

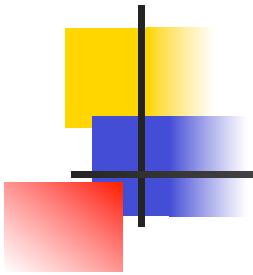
1. L'émetteur E diffuse un message M aux membres du groupe.
2. Un récepteur R met à jour son estampille à la réception de M. Il stocke (M, S_E) dans la liste des messages. Il envoie S_R à E.
3. L'émetteur E attend toutes les estampilles et calcule la valeur maximum S_{max} qu'il envoie aux membres du groupe.

- A la réception de S_{max}, chaque destinataire marque M comme délivrable et lui associe le numéro d'ordre S_{max}.
- Un message est délivré dès qu'il est marqué comme délivrable et dispose du plus petit numéro d'ordre dans la liste des messages
- Complexité : 2N ou 2(N-1)

Enquête généralisée

Ordre total ou causal ?





Diffusion par horloge vectorielle

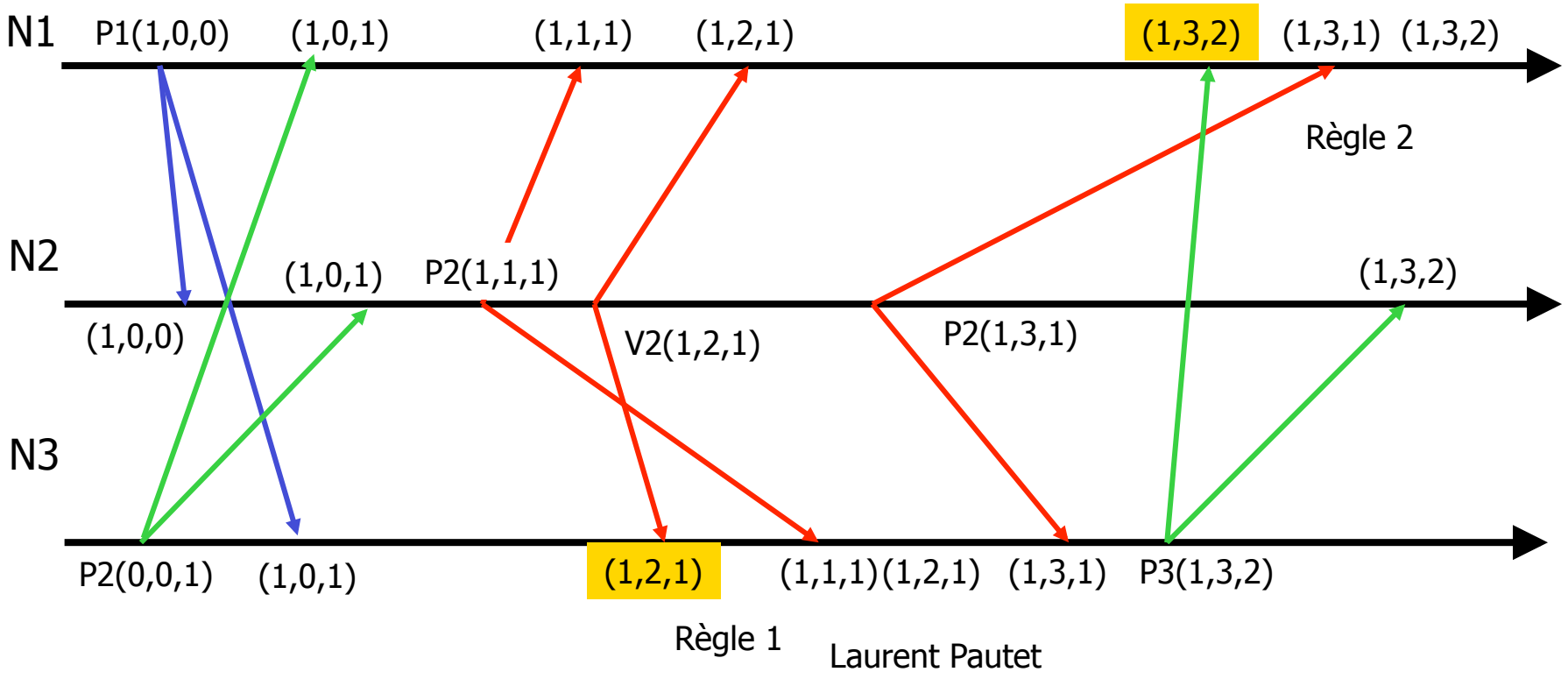
Ordre total ou causal ?

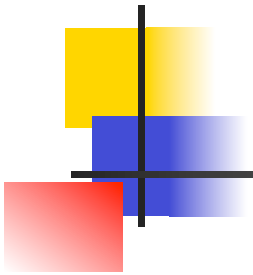
- Chaque nœud N_i initialise une horloge $V_i(1..N)$ à 0.
- Un nœud N_i incrémente $V_i(i)$ pour toute diffusion.
- Un nœud N_i estampille tout message M émis avec V_i.
- Le récepteur N_j délivre M estampillé V_m dès que
  1. Toutes les diffusions de N_i soient arrivées: $V_j(i) = V_m(i) - 1$.
  2. Toutes les diffusions antérieures à M et reçues sur N_i aient été aussi reçues par N_j : $\forall\ k \in 1..N$ et $k \neq i$, $V_j(k) \geq V_m(k)$
- Le récepteur N_j recale son horloge V_j . $\forall\ k \in 1..N$, $V_j(k) = \max(V_m(k),V_j(k))$

Diffusion par horloge vectorielle

Ordre causal mais pas total

- Pas d'ordre total : P1 et P2 sont concurrents
- Ordre causal : règles 1 et 2





Diffusion par horloge vectorielle

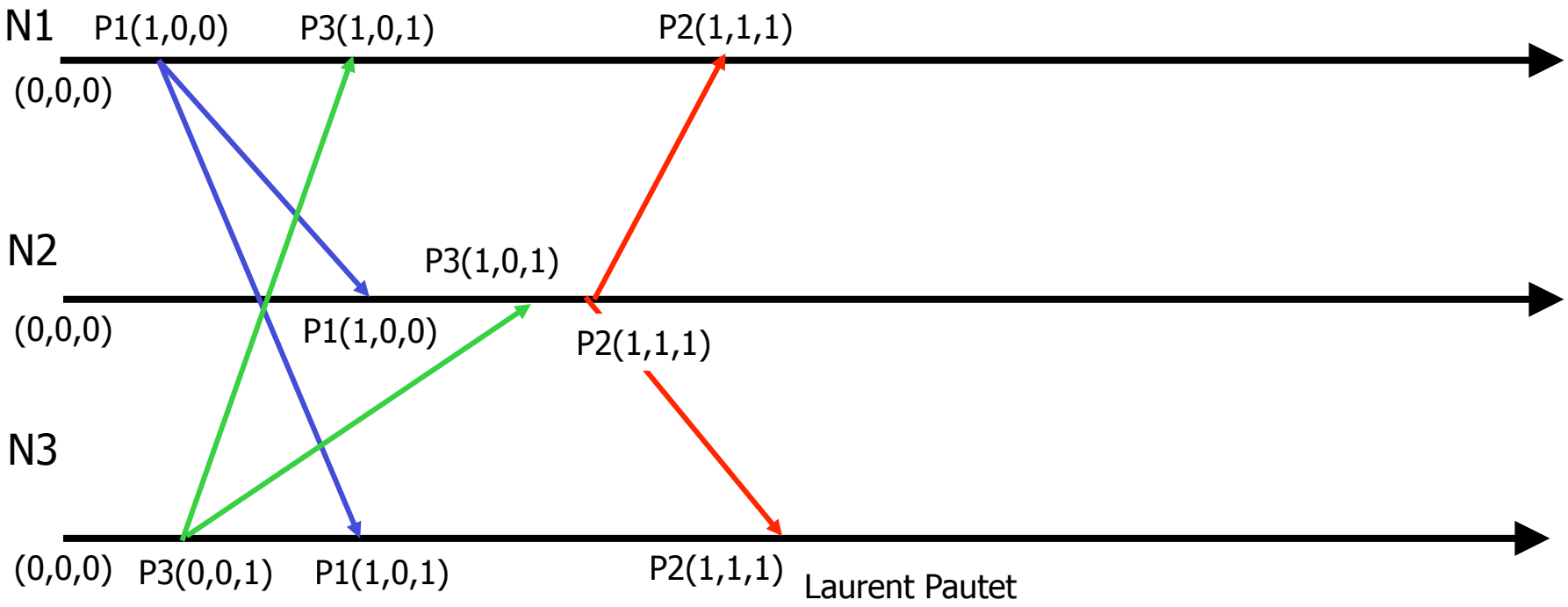
Comment obtenir un ordre total ?

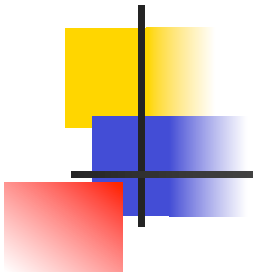
- Pour obtenir un ordre total, on peut reprendre l'approche de Ricart-Agrawal avec des horloges de Mattern (plus volumineuses que celles de Lamport)
- Une autre solution vise à forcer des diffusions régulières (vides) et à s'appuyer sur l'ordre causal
- Lors d'une modification de l'horloge de N_i, pour tout message M, émis par N_j, reçu et non-délivré N_i, si les conditions précédentes sont réunies et si $\forall\ k \in 1..N$, $V_M(j) \leq V_i(k)$ alors M peut être délivré
- Pour garantir l'ordre total, les messages qui sont délivrés alors, le sont par ordre croissant d'émetteurs

Diffusion par horloge vectorielle

Ordre causal total

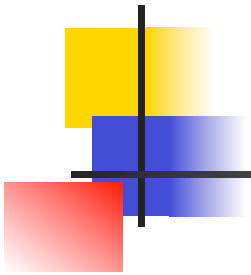
- A leur réception, P1 et P3 ne sont pas délivrés
- A la réception de P2, P1, P2 et P3 sont délivrés dans l'ordre croissant des identificateurs de nœuds





Exclusion mutuelle sans diffusion de groupe

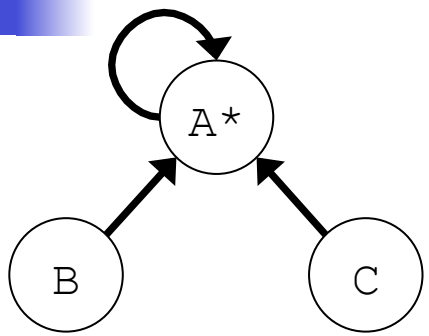
- Dans le cas de la diffusion, la file d'attente du verrou réparti est dupliquée et maintenue cohérente sur l'ensemble des nœuds grâce à l'ordre total causal
- Une autre approche consiste à répartir cette file d'attente du verrou entre les nœuds
- On va donc établir une liste des propriétaires successifs du verrou qui va se modifier dynamiquement
- Retrouver le dernier nœud ayant obtenu le verrou consiste donc à remonter la liste de propriétaires successifs



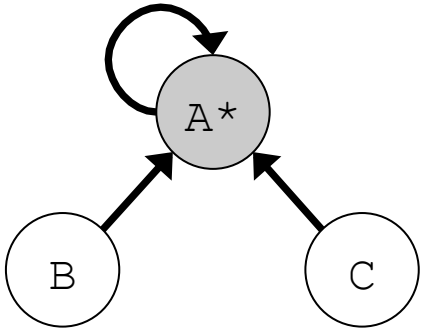
Algorithme de Naimi-Trehel

- Chaque agent tient à jour l'identité du dernier demandeur dont il a reçu la requête (lastOwner)
- Chaque agent tient à jour l'identité du demandeur auquel il a promis d'envoyer le jeton lorsqu'il en disposera (nextOwner)
- Lorsqu'il reçoit une requête pour obtenir le jeton
  - S'il détient le jeton, il note qu'il doit faire suivre celui-ci lorsqu'il n'en a plus besoin en mettant à jour lastOwner et nextOwner
  - Sinon, il fait suivre la demande au dernier destinataire (lastOwner) et met à jour celui-ci pour des demandes futures

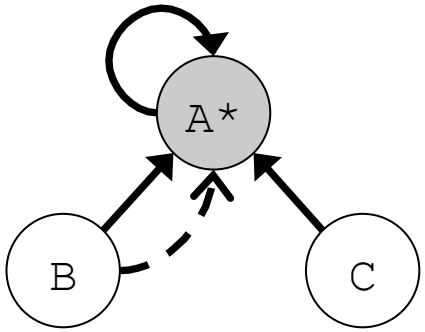
Scénario possible



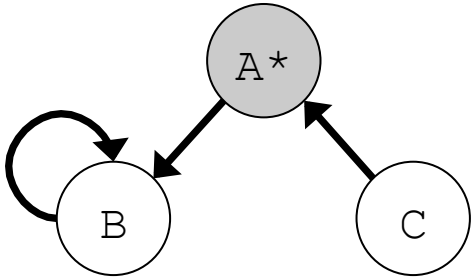
A possède O



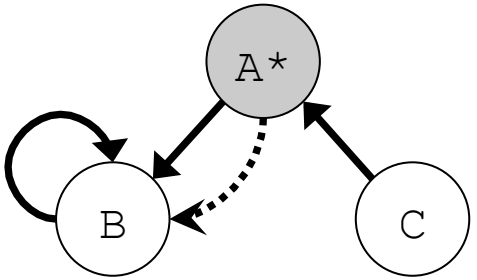
A modifie O



B veut modifier O



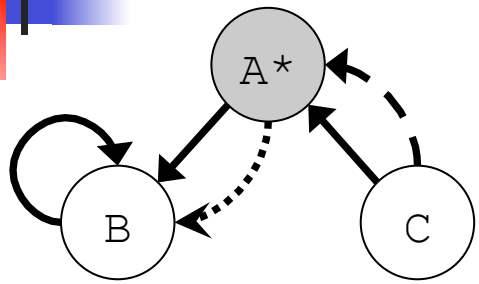
A et B désignent B
comme dernier propriétaire



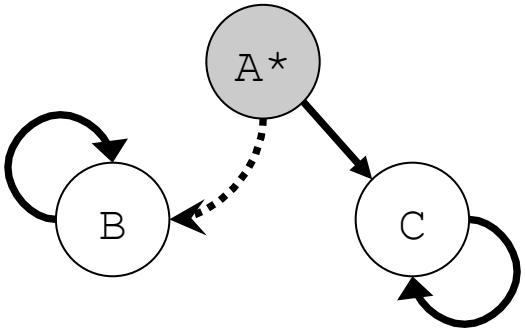
A désigne B comme
prochain propriétaire

→ lastOwner ⇢ nextOwner ⇢ requête

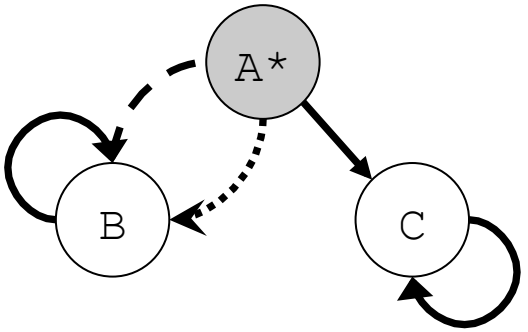
Scénario possible



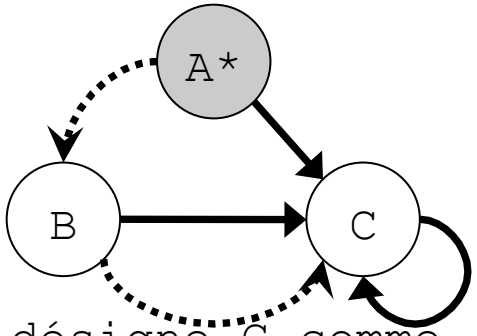
C veut modifier O et envoie sa demande à A



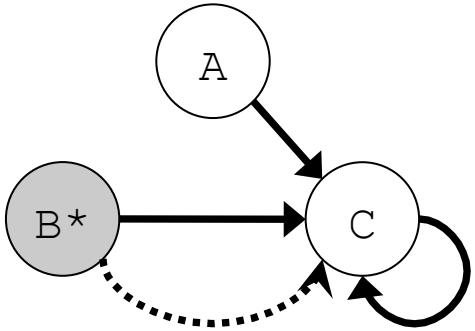
A et C désignent C comme dernier propriétaire



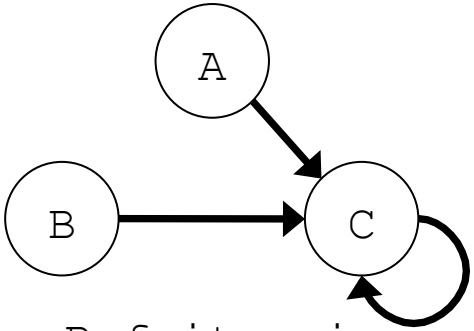
A fait suivre la demande de C à B



B désigne C comme prochain et dernier propriétaire



A fait suivre O à B



B fait suivre O à C

→ lastOwner

⇢ nextOwner

⇾ requête